Periodic Thermal Management for Hard Real-time Systems

Long Cheng, Kai Huang, Gang Chen, Biao Hu and Alois Knoll
Robotics and Embedded Systems
Technische Universitat Muenchen, Munich, Germany
Email: {chengl, huangk, cheng, hub, knoll}@in.tum.de

***Abstract*—Due to growing power density, on-chip temperature increases rapidly, which has hampered the reliability and performance of modern real-time systems. This paper studies how to minimize the peak temperature for hard real-time systems under hard real-time constraints with periodic thermal management . A closed-form representation of the peak temperature for such a periodic scheme is derived to tackle this problem. Based on this closed-form and the arrival curve model which is used to model the system workload, two approaches that can derive periodic thermal management are proposed to minimize the peak temperature for a given event stream with a trade-off between complexity and accuracy. Case studies show that our approaches can achieve similar or better level of peak temperature but with two or three orders of magnitude lower computation expense compared to previous work.**

I. Introduction

For a hard real-time system, its operation integrity strongly depends upon two essential principles [1]: the logical results of the computation must be correct and the time at which these results are produced must satisfy certain constraints. The timing constraints are usually specified as deadlines within which all the tasks should finish. Meeting deadlines is crucial for hard real-time systems, otherwise critical failures may be caused. It is often the case that hard real-time systems are employed in safety critical applications, such as an automated flight control system, an artificial heart, etc. In these cases, a small fault in the correctness of the system timing manner may cause extremely serious disasters. To ensure deadlines, real-time system designers need to consider an important factor, the temperature of the processor, which plays a key role in determining the allowable execution speed [2]. However, as the power density in current electronic devices is exponentially increasing, the temperature on current processors can be considerably high, which severely harms the reliability and performance of hard real-time systems.

To control the on-chip temperature, apart from using hardware cooling devices, alternative technologies which dynamically detect and alleviate thermal violations have also been adopted. Such technologies can be generally termed as Dynamic Thermal Management (DTM) techniques [3]. Since heat generation directly originates from power consumption, DTM techniques manage the temperature via controlling the power consumption of the processor. As power consumption of processors mainly comes from dynamic and leakage power consumption, DTM techniques follow two main mechanisms, i.e., Dynamic Voltage Frequency Scaling (DVFS) [4]–[8], in which the dynamic power consumption is reduced by adjusting the supply voltage or frequency of the processor such that the temperature is lowered, and Dynamic Power Management (DPM) [2], [9]–[11], which transits the processor to sleep state to diminish the leakage power such that the on-chip temperature gets controlled.

As transistor technology is shifting toward sub-micron domain, the leakage power increases exponentially and becomes comparable or even greater than dynamic power [12]. According to ITRS2011 [13], leakage power dominates the total power consumption of 32 nm or more advanced processors. Therefore, using DPM technologies to optimize the temperature on modern processors is more effective. In addition, DPM techniques can also be applied on peripheral devices such as memories, interconnects, etc. In this paper, we propose a DPM approach, which switches the processor between active and sleep modes according to certain patterns, to minimize the peak temperature.

The main issue of DPM technologies is when and how long one should turn the processor to the sleep state [14]. Meisner et al. [10] proposed a power management approach, called PowerNap, to eliminate idle power in servers by quickly transiting in and out of an ultra-low power state. The processors are activated from the ultra-low power state if an activity is detected in the network interface controller. However, erratically transiting the processor to a low power state causes predicting the thermal behavior almost impossible. Furthermore, since the performance of the processor is reduced when a DPM technology is applied, one should carefully evaluate the execution time penalty to ensure that all events finish within their deadlines. This evaluation is also difficult to complete due to the unpredictability in power state transition. To make it easy to predict the temperature evolution and execution time penalty of the processor, one can periodically put the processor in a lower power mode. Masud Ahmed et al. [2] presented an offline algorithm for sporadic tasks to minimize the peak temperature in embedded real-time systems by utilizing thermal-aware periodic resources . The sporadic model of the workload causes pessimism for the analysis and results in higher peak temperature of the system, as this model cannot model non-determinism like jitter or burst arrivals of the system workload. In view of this, our DPM technique designed for hard real-time systems should: (1) guarantee all events complete within deadlines, (2) work with general event arrival patterns, (3) explicitly demonstrate the pattern of power mode transition and how the temperature evolves, (4) minimize the peak temperature of the processor.

In this paper, we propose the periodic thermal management (PTM), which holds the properties mentioned above, to optimize the peak temperature for general events arrivals while the deadlines are guaranteed. A single core processor that has two power dissipation modes, ‘active’ and ‘sleep’ mode, is considered. The peak temperature is controlled by periodically switching the processor into the sleep mode. The PTM scheme is derived offline in order to achieve minimum runtime overhead. Real-time Calculus [15] is employed to model the non-deterministic event arrivals and service provided by the processor in the time interval domain. The detailed contributions of this paper are as follows:

- A closed-form solution of the peak temperature with respect to the periodic thermal management is developed.
- Two PTM algorithms that can derive periodic on/off schemes with a trade-off between accuracy and efficiency are developed. One offers precise solution by making thorough searches and the other is a fast approximation based on bounded-delay function.
- The effectiveness and efficiency of our algorithms are studied by comparison to two related work [2], [11] in the literature.

The rest of this paper is organized as follows. The related work is introduced in the next section. Section III presents system models, including hardware model, event model and thermal model, and the problem definition. Section IV derives the closed-form solutions of the peak temperature. Section V presents our PTM algorithms. Several cases are studied in Section VI and Section VII concludes this paper.

II. Related Work

As stated in previous section, the thermal behaviour of a processor is directly influenced by the power consumption. Thus researchers in previous work on thermal-aware scheduling have followed two main approaches: DVFS and DPM, which have already been widely exploited in power-aware scheduling. In this section, we overview previous work for thermal-aware scheduling that based on DVFS and DPM.

Sushu Zhang et al. [4] proposed two DVFS approaches: a pseudo-polynomial optimal algorithm and a fully polynomial time approximation one. These two approaches can optimally and approximately improve the system performance for a set of periodic tasks under thermal constraints, respectively. Jian-Jia Chen et al. [8] presented two approaches to schedule periodic real-time tasks under DVFS while the response time and temperature constraints are satisfied respectively. Chantem et al. [5] made an observation about maximizing the workload under thermal constraints. The authors demonstrated that while working with proactive scheduling, the scheduler which maximizes the workload under given peak temperature must be a periodic one [2]. According to this observation, a speed schedule was proposed to maximize the workload based on DVFS with discrete speeds and transition overhead under given temperature constraints. S. Wang et al. [6] presented a reactive speed control algorithm for tasks that have the same period to minimize temperature and performed several schedulability tests. The aforementioned work, however, based on either a simplified workload model, such as periodic tasks, or the processor feature of keeping the ‘ideal’ speed, which may not be found in recent top-of-the-line microprocessors [2]. The periodic thermal management (PTM) proposed in this paper can handle general event arrival patterns by adopting real-time calculus [15]. Moreover, lower power state, which is a basic power management feature, can be conveniently utilized to implement PTM.

There are also several researches that utilize DPM to minimize the peak temperature under deadline constraints. Kumar et al. [9] developed a thermally optimal stop-go scheduling called JUst Sufficient Throttling (JUST) to minimize peak temperature within given makespan constraints. This scheduling is designed only for static order tasks and is not applicable for non-deterministic tasks. To address the challenge of determining the real-time guarantees in the presence of unpredictable dynamic environmental conditions, Hettiarachchi and et al. [16] proposed a framework and mechanisms for thermal stress analysis in real-time systems. Adopting thermal-aware periodic resources, Masud Ahmed et al. [2] proposed an offline algorithm which minimizes the peak temperature for sporadic tasks scheduled by earliest-deadline first (EDF) while guaranteeing all their deadlines can be met. The workload models of the aforementioned work are also simplified and lead to pessimistic results, that is, higher peak temperature since they cannot exhibit non-determinism like jitter or burst arrivals of the workload. These shortcomings can also be overcome in PTM since it work with general event arrival patterns, as mentioned above. In [11], a Cool Shaper is studied to minimize the peak temperature by delaying the execution of workload for general events arrivals. It is an online/offline-combined approach, where the parameters of the shaper are offline computed and the workload is runtime orchestrated with the pre-computed shaper. Besides the online monitoring overhead which can result in a higher temperature, determining the parameters of the shaper according to the system specification also requires considerable calculation effort. In this paper, a closed form of the peak temperature is derived such that our PTM can easily obtain the peak temperature offline instead of simulating the online evolution of the temperature, which saves great quantity of calculation.

III. System Model

A. Hardware Model

A single core processor that has two power dissipation modes, i.e., ‘active’ and ‘sleep’ mode, is adopted in this paper. The processor must be in ‘active’ mode with a fixed speed to process coming event streams with power consumption P_a and can be turned to ‘sleep’ mode with a lower power consumption P_s when there is no event to handle. We consider the time and power overheads during model-switching. t_{swoff} and t_{swon} time units are required to switch the processor from ‘active’ mode to ‘sleep’ mode and back, respectively. During mode switching, the power dissipation equals P_a but the processor does not tackle any coming event. The time and power overheads during mode switching have nontrivial impacts on the resource providing capability and thermal evolution of the processor. For example, suppose the processor is switched to ‘active’ mode first and then t_{on} time units later it is turned to ‘sleep’ mode and stays at this mode for t_{off} time units. As shown in Fig. 1, in this $(t_{on} + t_{off})$ units time interval, the length of the overall time slots in which the processor can handle coming events is shorter than t_{on} and the time interval during which the processor consumes power P_s is shorter than t_{off}. Therefore, the mode-switching overhead leads to a higher temperature and a weaker resource providing capability. The quantitative impacts will be investigated later. Moreover, as shown in Fig. 1, the time lengths for which the processor is

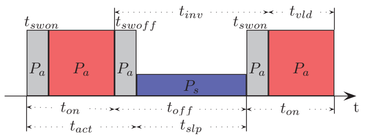

Fig. 1. Hardware model.

switched to ‘active’ and ’sleep’ mode must be larger than t_{swon} and t_{swoff}, respectively:

$$t_{off} > t_{swoff} \quad , \quad t_{on} > t_{swon}. \tag{1}$$

B. Event Model

Without loss of generality, we assume that the input events arrive at the system irregularly. In order to model workload of such coming events, the concept of arrival curve in network calculus [17] and real- time calculus [15] is adopted. Before introducing arrival curve, we briefly present the concept of cumulative workload $R(t)$. For a single event stream, the cumulative workload $R(t)$ represents the number of events arrive at the processor in time interval $[0,t)$ with $R(0) = 0$ by convention. According to its definition, $R(t)$ always specifies one concrete trace of event stream. Therefore, arrival curve $\bar{\alpha}(\Delta) = [\bar{\alpha}^u(\Delta), \bar{\alpha}^l(\Delta)]$, which can bound admissible traces of event streams, is introduced to provide an abstract model for event streams. The upper arrival curve $\bar{\alpha}^u(\Delta)$ and the lower arrival curve $\bar{\alpha}^l(\Delta)$ are the upper and lower bound of $R(t)$:

$$\bar{\alpha}^l(\Delta) \le R(t-s) \le \bar{\alpha}^u(\Delta), \forall t-s = \Delta \text{ and } s \ge 0 \tag{2}$$

where $\bar{\alpha}^l(0) = \bar{\alpha}^u(0) = 0$. With the concept of arrival curve, we can unify many other common timing models of event stream. For example, a periodic event stream can be abstracted by a set of step function where $\bar{\alpha}^u(\Delta) = \left\lfloor \frac{\Delta}{p} \right\rfloor + 1$ and $\bar{\alpha}^l(\Delta) = \left\lfloor \frac{\Delta}{p} \right\rfloor$. A sporadic event stream can also be modeled by $\bar{\alpha}^u(\Delta) = \left\lfloor \frac{\Delta}{p} \right\rfloor + 1$, $\bar{\alpha}^l(\Delta) = \left\lfloor \frac{\Delta}{p'} \right\rfloor$, where p and p' are the minimal and maximal inter arrival distance of the event stream, respectively. Moreover, for an event stream which can be specified by a period p, jitter j and minimal inter arrival distance d, the upper arrival curve is $\bar{\alpha}^u(\Delta) = \min\{\left\lceil \frac{\Delta+j}{p} \right\rceil, \lceil \frac{\Delta}{d} \rceil\}$.

To model the resource providing capability, we adopt service curve $\beta(\Delta) = [\beta^u(\Delta), \beta^l(\Delta)]$ analogously [17]. In the same way, service curve provides the upper bound and lower bound of cumulative function $C(t)$, which is the number of total time slots that the processor provides to handle coming events in time interval $[0,t)$. The upper service curve $\beta^u(\Delta)$ and lower service curve $\beta^l(\Delta)$ satisfy:

$$\beta^l(\Delta) \le C(t-s) \le \beta^u(\Delta), \forall t-s = \Delta \text{ and } s \ge 0 \tag{3}$$

where $\beta^l(0) = \beta^u(0) = 0$.

It is worth noting that the arrival curve $\bar{\alpha}(\Delta)$ is event-based and specifies the upper and lower bounds of the number of input events in any time interval Δ, while the service curve $\beta(\Delta)$ is time-based and specifies the upper and lower bounds of the amount of available execution time in any time interval Δ. Thus, we transform the event-based arrival curve $\bar{\alpha}(\Delta)$ to time-based arrival curve $\alpha(\Delta)$, which describes the amount of execution time demanded by arrival events. Suppose that the worst-case execution time of one event in arrival stream is c, then the arrival curve transformation can be performed as $\alpha^u(\Delta) = c \cdot \bar{\alpha}^u(\Delta)$ and $\alpha^l(\Delta) = c \cdot \bar{\alpha}^l(\Delta)$ [18].

Our PTM approaches are designed to handle not only single event streams but also multi-event streams. For multi-event scenarios, N event streams are supposed in the input source, where $N \ge 2$. We order the event streams $S_1, S_2, \cdots, S_N$ according to their relative deadlines, where D_i, the relative deadline of event stream S_i, is smaller than that of S_j when $i < j$. Thus, the input event model of our processor can be depicted by the tuple $\mathbf{EM}(N) = (\bar{\alpha}(\Delta)_1, c_1, D_1, \cdots, \bar{\alpha}(\Delta)_N, c_N, D_N)$. Now, a processor with service curve $\beta(\Delta)$ satisfies the worse-case deadline constraints for its workload, which is modeled by $\mathbf{EM}(N)$, when the following condition holds:

$$\beta^l(\Delta) \ge \beta_B(\Delta), \forall \Delta \ge 0. \tag{4}$$

where $\beta_B(\Delta) = \alpha^u(\Delta - D)$ for single event scenarios. For multi-event scenarios, $\beta_B(\Delta)$ has different formulations with different scheduling policies, which is detailed later.

C. Thermal Model

In this paper, the temperature model of the processor is based on the Fourier law of heating [19], which can be described by the following equation:

$$C\frac{dT}{dt} = P - G(T - T_{amb}) \tag{5}$$

where T, C, G and P denote the temperature, thermal capacitance, thermal conductance and power dissipation of the system, respectively. T_{amb} indicates the ambient temperature. In addition, the absolute temperature (Kelvin, K) is set as the unit of all temperature variables. We assume the power dissipation has a linear relationship with respect to temperature in this paper [19]. Thus the power dissipation is formulated as:

$$P = \varphi T + \theta \tag{6}$$

where φ and θ are constants. Rewriting (5), we have

$$\frac{dT}{dt} = -mT + n \tag{7}$$

where $m = \frac{G-\varphi}{C}, n = \frac{\theta + GT_{amb}}{C}$. Then the the steady-state temperature of currently working state can be derived as $T^\infty = n/m$ with $\frac{dT}{dt} = 0$. As m and n are constants, a closed-form solution of the temperature yields:

$$T(t) = T^\infty + (T_{init} - T^\infty) \cdot e^{-m \cdot t} \tag{8}$$

where T_{init} indicates the initiate temperature. As mentioned in hardware model, the processor has two power dissipation modes and consumes power P_a during mode-switching. Therefore when the processor is in mode-switching or ‘active’ mode, the power consumption is $P_a = \varphi_a T(t) + \theta_a$, otherwise the power consumption is $P_s = \varphi_s T(t) + \theta_s$ ($<\varphi_a, \theta_a>$ and $<\varphi_s, \theta_s>$ are the corresponding parameters in Eqn. 6). Thence, the coefficient for Eqn. 8 are given as [11], [19]:

$$\begin{aligned} m_a &= \frac{G - \varphi_a}{C} \quad , \quad m_s = \frac{G - \varphi_s}{C} \\ T_a^\infty &= \frac{\theta_a + GT_{amb}}{G - \varphi_a} \quad , \quad T_s^\infty = \frac{\theta_s + GT_{amb}}{G - \varphi_s} \end{aligned} \tag{9}$$

In addition, we also regulate the thermal model by these following circumstances.

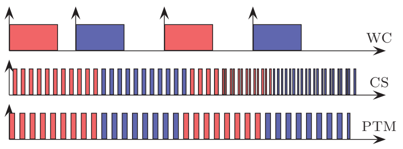

Fig. 2. Execution of jobs in policy WC, DT and PTM.

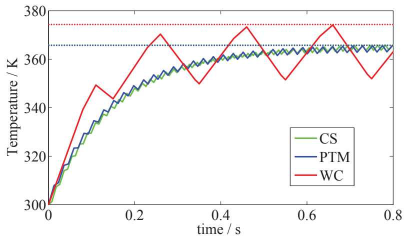

Fig. 3. Temperature evolution in policy WC, DT and PTM.

- $m_a > 0$ and $m_s > 0$.
- The steady-state temperature in 'active' mode is non-smaller than the one in 'sleep' mode, that is, $T_a^\infty \geq T_s^\infty$.
- The initial temperature $T_{init} = T_{amb} \leq T_s^\infty$.

Thus, the thermal mode of the processor in this paper is characterized by the tuple $\mathbf{TM} = (T_a^\infty, m_a, T_s^\infty, m_s)$.

D. Problem Statement

Dynamically switching the processor into 'sleep' mode according to the event arrivals is an effective way to minimize the peak temperature. However, this needs vast calculating efforts, which hampers the efficiency. Periodic thermal management (PTM), a trade-off between effect and efficiency, is adopted in this paper to minimize the peak temperature by periodically putting the system into 'active' and 'sleep' modes. In each period, the processor stays at 'active' mode and 'sleep' mode for t_{on} and t_{off} time units, respectively. In addition, $t = t_{on} + t_{off}$ denotes the length of the period. We illustrate our approach with an example in which three thermal management policies are adopted: (a) a work conserving (WC) execution that with no DTM policy, which means that the processor stays at 'active' mode to process events if there is (at least) one event in the ready queue, (b) an online DPM policy called Cool Shaper (CS) which dynamically transits the processor into 'sleep' mode according to the event arrivals, and (c) periodic thermal management (PTM). The thermal and hardware parameters are described in Tab. I. The event stream is set as: period $p = 0.2s$, jitter $j = 0.05s$, minimal inter-arrival distance $d = 0.001s$, execution time $c = 0.11s$ and relative deadline $D = 1.2p$. A concrete trace of events that arrive at time $(0, 0.15, 0.35, 0.55)s$ is adopted in this example. Fig. 2 and Fig. 3 show the execution of events and the temperature evolution for the three policies, respectively. As shown in Fig. 3, the peak temperature in policy PTM is slimly higher than the one in policy CS and they are both about 9 K less than the one in policy WC. This indicates that PTM policy can achieve close results to CS policy in terms of peak temperature and they are both effective compared to WC policy. From Fig. 2, we find that PTM can be seen as an approximate policy of CS, this interprets why the peak temperature of PTM is slimly higher. Despite of this, PTM requires less resources for computation with acceptable results and is very convenient to implement.

This paper considers the temperature varying in a time interval L, where $L >> t$ and L/t is an integer. Due to the model-switching overhead, t_{on} and t_{off} cannot be directly utilized into thermal mode and service curve. Before giving the revised solutions, we first define some notations. From Fig. 1, t_{act} and t_{slp} denote the time interval that the processor consumes power P_a and P_s in one period, respectively. Analogously, t_{vld} denotes the time interval that the processor can tackle coming events in one period and t_{inv} represents the rest. Based on hardware model, we formulate them as:

$$t_{act} = t_{on} + t_{swoff} \quad , \quad t_{slp} = t_{off} - t_{swoff} \tag{10}$$

$$t_{vld} = t_{on} - t_{swon} \quad , \quad t_{inv} = t_{off} + t_{swon}. \tag{11}$$

With these definitions, one can use t_{act} and t_{slp} to derive the peak temperature and t_{vld} and t_{inv} to calculate the service curve of the processor; meanwhile, the time and power overhead of mode-switching are considered.

Now we define our problem as follows:
Given a system characterized by the power model and the thermal model ***TM*** *described in the preceding pages, task streams that are modeled by* ***EM***(N)*, our goal is to derive a periodic thermal management depicted by* t_{on} *and* t_{off} *such that the peak temperature is minimized while all the events complete within their deadlines.*

IV. Peak Temperature Analysis

In this section, we derive the formula of the peak temperature in PTM such that our algorithm can utilize it as a criterion of the optimal pair of $< t_{on}, t_{off} >$.

As PTM periodically transits the system between two power modes, the values of the parameters in the temperature model Eqn. 8 change periodically, which causes the general solution of the transient temperature T very complicated. Therefore, instead of utilizing the general solution, we derive the formula of the peak temperature based on some basic lemmas, which are obtained from close observations of the temperature evolution and are presented in the following.

Lem. 1: With a periodic thermal management PTM (t_{on}, t_{off}), the temperature of the processor ceaselessly rises in the opening few periods and then rises in t_{act} and descends in t_{slp} in every following period.

Proof: As mentioned before, the initial temperature $T_{init} = T_{amb} \leq T_s^\infty \leq T_a^\infty$. Based on Eqn. 7, inequality $\frac{dT}{dt} \geq 0$ holds in the beginning several periods when $T \leq T_s^\infty$. Therefore, temperature T continuously rises and then reaches T_s^∞. It's worth noting that T will never surpass T_a^∞ unless the initial temperature is higher than T_a^∞, as T_a^∞ is the steady-state temperature of the 'active' mode. Since T has already passed T_s^∞, it also will never drop back below T_s^∞ until being shut down. Therefore, one can summarize that PTM will keep the temperature T change between T_s^∞ and T_a^∞ once T passes T_s^∞. Again, based on Eqn. 7, one can observe that $\frac{dT}{dt} \geq 0$ in t_{act} and otherwise $\frac{dT}{dt} \leq 0$. ■

Based on Lem. 1, in the jth period, the temperature T reaches its local maximum T_j at the end of the time interval t_{act}. The peak temperature $T^\star$ can be defined as the maximum of all the T_j:

$$T^\star = \max(T_1, \cdots, T_{\frac{L}{t}}). \tag{12}$$

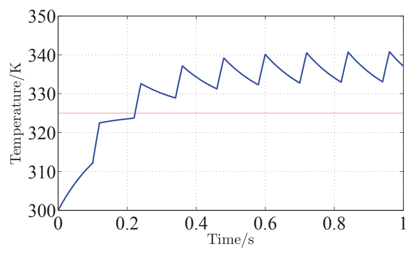

Fig. 4. Example of temperature varying with PTM($t_{on} = 0.02s, t_{off} = 0.1s$) while the model-switching overhead is not considered. The thermal and hardware parameters are described in Tab. I.

As shown in Fig. 4, the local maximum increases in the beginning and then stays at a stable value in the rest time. This reveals that the peak temperature can be obtained based on the difference between two consecutive local maximums, which is depicted in the following lemma.

Lem. 2: Denoting the local maximal temperature in the jth period as T_j, the temperature difference between two consecutive local maximums, $T_{j+1} - T_j$, can be formulated as:

$$\begin{aligned} T_{j+1} - T_j =& (1 - e^{-m_a t_{act}})T_a^\infty + e^{-m_a t_{act}}(1 - e^{-m_s t_{slp}})T_s^\infty \\ &- [1 - e^{-m_a t_{act} - m_s t_{slp}}]T_j \end{aligned} \tag{13}$$

where t_{act} and t_{slp} are from Eqn. 10.

Proof: With T_j, t_{slp} and Eqn. 8, we first derive T_j^{slp}, which is the temperature at the end of t_{slp} in the jth period. From Eqn. 8, one can get $T_j^{slp} = T_s + (T_j - T_s)e^{-m_s t_{slp}}$. Then with t_{act}, T_a and T_j^{slp}, Eqn. 8 generates the following equation:

$$\begin{aligned} T_{j+1} =& (1 - e^{-m_a t_{act}})T_a^\infty + e^{-m_a t_{act}}(1 - e^{-m_s t_{slp}})T_s^\infty + \\ & e^{-m_a t_{act} - m_s t_{slp}} T_j. \end{aligned} \tag{14}$$

Subtracting T_j from both sides yields Eqn. 13. ■

With above lemmas, the first main result of this paper is presented as the theorem below:

Thm. 1: Given a system as stated above and a periodic thermal management PTM (t_{on}, t_{off}), the peak temperature of the processor is a linear combination of T_a^∞ and T_s^∞, which is given as:

$$T^\star = \lambda T_a^\infty + (1 - \lambda)T_s^\infty, \tag{15}$$

where

$$\lambda = \frac{1 - e^{-m_a t_{act}}}{1 - e^{-m_a t_{act} - m_s t_{slp}}}.$$

Proof: We prove Thm. 1 by contradiction. For brevity, $\lambda T_a^\infty + (1-\lambda)T_s^\infty$ is denoted as $T^\diamond$. First, suppose that the peak temperature $T^\star$ is reached in the ith period and $T_i = T^\star < T^\diamond$. Rewriting Eqn. 13 yields that $T_{i+1} - T_i > 0$, which contradicts to the presumption that T_i is the peak temperature of the processor.

Similarly, assume that the peak temperature $T^\star$ is reached in jth period and $T_j = T^\star > T^\diamond$. Therefore we have:

$$T_j - T_{j-1} > T^\diamond - T_{j-1} \tag{16}$$

According to Lem. 2:

$$\begin{aligned} T_j - T_{j-1} &= (1 - e^{-m_a t_{act} - m_s t_{slp}})[\lambda T_a^\infty + (1-\lambda)T_s^\infty - T_{j-1}] \\ &= (1 - e^{-m_a t_{act} - m_s t_{slp}})(T^\diamond - T_{j-1}). \end{aligned}$$

Since $(1 - e^{-mt}) < 1$, the following inequality yields:

$$T_j - T_{j-1} < T^\diamond - T_{j-1} \tag{17}$$

which is in conflict with Eqn. 16. In conclusion, $T^\star = T^\diamond$. ■

Next, the boundaries of t_{on} and t_{off} are explored, then two approaches are proposed to minimize T^*.

V. PTM Algorithms

A. Service Bound of PTM

Real-time interface is employed in this paper to ensure that all events complete within their deadlines. With the hardware model described before and a given PTM (t_{on},t_{off}), the lower service curve of the processor is written as:

$$\beta_R(\Delta) = \max\left(\left\lfloor \frac{\Delta}{t} \right\rfloor \cdot t_{vld}, \Delta - \left\lceil \frac{\Delta}{t} \right\rceil \cdot t_{inv}\right), \tag{18}$$

where t is the period, t_{vld} and t_{inv} are obtained from Eqn. 11. To satisfy the deadline constraints, the lower service curve of the processor $\beta_R(\Delta)$ should satisfy the following inequality:

$$\beta_R(\Delta) \geq \beta_B(\Delta),\ \forall \Delta \geq 0, \tag{19}$$

where $\beta_B(\Delta)$ is the service bound for the workload modeled by **EM**(N). For a single event stream ($N = 1$), $\beta_B(\Delta)$ can be formulated as [18], [20]:

$$\beta_B(\Delta) = a^u(\Delta - D) \tag{20}$$

For multi-event streams ($N \geq 2$), the service bound $\beta_B(\Delta)$ in Eqn. 19 should be computed based on the scheduling policy. Note that only the service bound $\beta_B(\Delta)$ has to be revised. The other parts of our algorithms can remain untouched. Suppose the scheduling policy of earliest deadline first (EDF) is adopted, the service bound for the N event streams is [18]:

$$\beta_B(\Delta) = \sum_{i=1}^{N} \alpha_i^u(\Delta - D_i). \tag{21}$$

It's worth noting that EDF is not necessarily the only one scheduling policy can be adopted here. For example, when fixed priority (FP) scheduling is employed, the service bound can be calculated according to another formula [20] and fits in with our algorithms as suitable as EDF.

B. Feasible Region of t_{off}

In this paper, our goal is to find the optimal $< t_{on}, t_{off} >$ under the deadline constraints. Apparently brutal searching the whole two-dimensional space is the least efficient way to find the solution. Based on Eqn. 15, one can find that the derivative of $T^\star$ with respect to t_{on} is $\frac{\mathrm{d}T^\star}{\mathrm{d}t_{on}} = (T_a^\infty - T_s^\infty)\frac{m_a e^{-m_a(t_{on}+t_{swoff})}[1-e^{-m_s(t_{off}-t_{swoff})}]}{[1-e^{-m_a t_{act} - m_s t_{slp}}]^2} > 0$. Therefore, for a given t_{off}, $T^\star$ can be minimized by searching the minimal t_{on} under the service curve constraint, Eqn. 19. Based on this feature, our algorithms generally include two steps: (1) finding the minimal t_{on} for a given t_{off} with the real-time constraint, and (2) varying t_{off} in its feasible region and getting the optimum value when $T^\star$ reaches its minimum.

In order to discover the minimal t_{on}, the feasible region of t_{off} should be determined first such that one can assure the solution to the minimal t_{on} exists. For example, when the input is a single event stream and $t_{off} = D$, coming events in worst-case will miss their deadlines before they are processed, considering additional t_{swon} time units are required to switch the processor on. According to the hardware model, we directly know that t_{off} has to be no less than t_{swoff} to cover the timing overhead of model-switching. To avoid situations similar to the example, t_{off} must be bounded by an upper bound, which is calculated according to the maximum service curve in [18]:

$$t_{off}^{max} = \max\left\{t_{off} : \beta_R^{\top}(\Delta) \geq \beta_B(\Delta), \forall\Delta \geq 0\right\}, \qquad (22)$$

where $\beta_R^{\top}(\Delta)$ can be formulated as follows when we take t_{swon} into account:

$$\beta_R^{\top}(\Delta) = \max\{0, \Delta - t_{off} - t_{swon}\} \qquad (23)$$

Finally, the feasible region of t_{off} can be depicted as $t_{off} \in [t_{swoff}, t_{off}^{max}]$.

C. Searching the minimal t_{on}

Based on the constraint Eqn. 19, when t_{off} is fixed, the precise solution of minimal t_{on} can be calculated as:

$$t_{on}^{prc} = \min\left\{t_{on} : \beta_R(\Delta) \geq \beta_B(\Delta), \forall\Delta \geq 0\right\}. \qquad (24)$$

This solution can be found by testing the t_{on}s starting from t_{swon} with step ε until the minimal t_{on} satisfying Eqn. 19 is discovered. By this method, the minimal t_{on} is obtained with high accurateness while the time consumption is significant.

To reduce the computational overhead, we adopt the bounded-delay function [18], [21] to calculate an approximate minimal t_{on}. As shown in Fig. 5, for a given t_{off}, this approach first finds an affine-line which originates from point (t_{off}, 0) and is also tangent to the service bound $\beta_B(\Delta)$. The affine-line is colored red in the figure. The slope of the tangent is denoted as $\eta(t_{off})$ and is defined by t_{off}:

$$\eta(t_{off}) = \inf\{\rho : \rho(\Delta - t_{off}) \geq \beta_B(\Delta), \forall\Delta \geq 0\} \qquad (25)$$

Then, the bounded-delay function $bdf(\Delta, \eta(t_{off}), t_{off})$ is formulated as:

$$bdf(\Delta, \eta(t_{off}), t_{off}) = \max[0,\ \eta(t_{off})(\Delta - t_{off})] \qquad (26)$$

Therefore, as shown in Fig. 5, $\beta_R(\Delta) \geq \beta_B(\Delta)$ holds as long as the lower service curve of processor is no less than $bdf(\Delta, \eta(t_{off}), t_{off})$.

When the mode-switching overhead is ignored, the approximate minimal t_{on} can be calculated as $t_{on}^{apx} = \frac{\eta(t_{off}) \cdot t_{off}}{1 - \eta(t_{off})}$ (Refer to Fig. 5 for the derivation). Since we take the time overhead into account, this equation is revised as $t_{vld}^{apx} = \frac{\eta(t_{inv}) \cdot t_{inv}}{1 - \eta(t_{inv})}$. Based on Eqn. 11, the revised approximate t_{on} is denoted as:

$$\begin{aligned} t_{on}^{apx} &= t_{vld}^{apx} + t_{swon} & (27)\\ &= \frac{\eta(t_{off} + t_{swon})}{1 - \eta(t_{off} + t_{swon})} \cdot (t_{off} + t_{swon}) + t_{swon} \\ &= \frac{\eta(t_{off} + t_{swon})}{1 - \eta(t_{off} + t_{swon})} \cdot t_{off} + \frac{t_{swon}}{1 - \eta(t_{off} + t_{swon})}. \end{aligned}$$

For brevity, we define function $RVT(\eta, t_1, t_2) = \frac{\eta \cdot t_1}{1-\eta} + \frac{t_2}{1-\eta}$. Therefore, the revised approximate t_{on} can be denoted as:

$$t_{on}^{apx} = RVT(\eta(t_{off} + t_{swon}),\ t_{off},\ t_{swon}) \qquad (28)$$

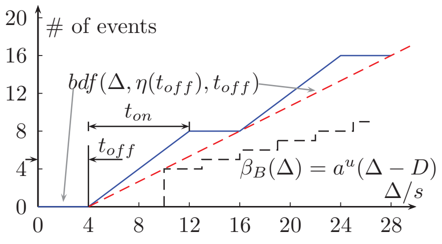

Fig. 5. Obtaining approximate minimal t_{on} based on the bounded-delay function when the mode-switching overhead is not considered.

D. Minimize the peak temperature

Based on the precise and the approximate solutions of minimal t_{on}, two algorithms with different accuracy and efficiency are presented to minimize $T^{\star}$, namely PMPT (precisely minimizes the peak temperature) and AMPT (approximately minimizes the peak temperature).

Algorithm PMPT For a given t_{off}, the minimal t_{on} in this algorithm is calculated based on Eqn. 24, which derives a precise solution. With t_{on}^{prc}, the corresponding peak temperatures $T^{\star}$ of all the tested t_{off}s can be computed from Thm. 1, then the minimal $T^{\star}$ in all the tested points indicates the optimal solution of t_{off}. Since there are several local minima of $T^{\star}$ and $T^{\star}$ has a irregular relationship with t_{off}, a thorough search with a fixed step ε in the feasible region of t_{off} is made to find the global minimal $T^{\star}$.

Algorithm 1 PMPT

```
Input: TM, EM($N$), $t_{swon}$, $t_{swoff}$, $\varepsilon$, $\xi$
Output: $t_{on}^{opt}$, $t_{off}^{opt}$
 1: calculate $\beta_B(\Delta)$ based on EM($N$) and the scheduling
    policy
 2: get $t_{off}^{max}$ from Eqn. 22
 3: $T_{min}^{\star} = T_a^{\infty}$, $t_{on}^{opt} = 0$, $t_{off}^{opt} = 0$
 4: for $t_{off} = t_{off}^{l}$ to $t_{off}^{u}$ with step $\varepsilon$ do
 5:     get $t_{inv} = t_{off} + T_{swon}$ from Eqn. 11
 6:     find $t_{on}^{min}$ by testing Eqn. 24 with step $\xi$
 7:     compute $t_{act}^{min}$ and $t_{slp}$ by Eqn. 10
 8:     $T^{\star}(t_{act}^{min}, t_{slp}) = \lambda(t_{act}^{min}, t_{slp})T_a^{\infty} + [1 - \lambda(t_{act}^{min}, t_{slp})]T_s^{\infty}$
 9:     if $T^{\star}(t_{act}^{min}, t_{slp}) < T_{min}^{\star}$ then
10:         $t_{on}^{opt} \leftarrow t_{on}^{min}$, $t_{off}^{opt} \leftarrow t_{off}$
11:         $T_{min}^{\star} \leftarrow T^{\star}(t_{act}^{min}, t_{slp})$
12:     end if
13: end for
```

Algorithm 1 outlines the pseudo-code of algorithm PMPT. It takes as input the thermal model **TM**, the input event model **EM**(N), the time overheads of mode-switching and the accuracy coefficients ε and ξ. The service bound is obtained based on the input and scheduling policy (line 1) and then the upper bound of t_{off} is generated (line 2). Then the optimal solution and minimal $T^{\star}$ are initialized in line 3. Lines 4-13 iteratively discover the t_{on}^{prc} and calculate the peak temperature $T^{\star}$ for all t_{off}s in the feasible region with a step ε. The t_{inv} is computed by Eqn. 11 to derive the lower service curve (line 5). Then t_{on}^{prc} is found by examining every candidate of t_{on} from the lower bound, t_{swon}, with a step ξ (line 6). Afterwards, t_{on}^{prc} and t_{off} are revised based on Eqn. 10 to derive the peak temperature (line 7). The peak temperature is calculated based on Thm. 1 (line 8) and then compared to $T^{\star}_{min}$. If the newly derived one is lower, the corresponding $< t_{on}, t_{off} >$ and $T^{\star}$ are assigned to the optimal solution and $T^{\star}_{min}$, respectively (lines 9-12).

Algorithm AMPT In this algorithm, the minimal t_{on} is obtained directly from the approximation in Eqn. 28 with less computation. Then according to Thm. 1, the peak temperature can be formulated as a function of t_{off}:

$$\begin{aligned} T^{\star} &= PT(t_{off}) \\ &= T_s^{\infty} + \frac{1 - a \cdot e^{\frac{-m_a t_{swon}}{1-\eta}} \cdot e^{\frac{-m_a \eta}{1-\eta} t_{off}}}{1 - b \cdot e^{\frac{-m_a t_{swon}}{1-\eta}} \cdot e^{(m_s - \frac{-m_a \eta}{1-\eta}) t_{off}}} (T_a^{\infty} - T_s^{\infty}) \end{aligned} \tag{29}$$

where $a = e^{-m_a t_{swoff}}, b = e^{(m_s - m_a) t_{swoff}}$ and $\eta = \eta(t_{off} + t_{swon})$. Based on a set of systemic experiments (the details are included in appendix), we conjecture that $PT(t_{off})$ is a unimodal function which has only one minimum in the feasible region of t_{off}. Therefore the gold section search can be utilized to find the optimal t_{off} instead of searching all t_{off}s exhaustively. The pseudo-code is detailed in Algorithm 2.

Algorithm 2 AMPT

```
Input: TM, EM($N$), $t_{swon}$, $t_{swoff}$, $\varepsilon$
Output: $t_{on}^{opt}$, $t_{off}^{opt}$
 1: calculate $\beta_B(\Delta)$ based on EM($N$) and the scheduling policy
 2: get $t_{off}^{max}$ from Eqn. 22
 3: $t_{off}^{b} = t_{swoff}$, $t_{off}^{e} = t_{off}^{max}$
 4: $t_{off}^{a} \leftarrow f_1(t_{off}^{s}, t_{off}^{e})$, $t_{off}^{b} \leftarrow f_2(t_{off}^{s}, t_{off}^{e})$    ▷ Two tested points selected by gold section
 5: $T_1^{\star} \leftarrow PT(t_{off}^{a})$, $T_2^{\star} \leftarrow PT(t_{off}^{b})$
 6: while $t_{off}^{e} - t_{off}^{a} > \varepsilon$ do
 7:     if $T_1^{\star} > T_2^{\star}$ then
 8:         $t_{off}^{2} \leftarrow t_{off}^{b}$, $t_{off}^{s} \leftarrow t_{off}^{a}$, $t_{off}^{a} \leftarrow t_{off}^{b}$
 9:         $t_{off}^{b} \leftarrow f_2(t_{off}^{s}, t_{off}^{e})$, $T_1^{\star} \leftarrow T_2^{\star}$, $T_2^{\star} \leftarrow PT(t_{off}^{b})$
10:     else
11:         $t_{off}^{2} \leftarrow t_{off}^{a}$, $t_{off}^{e} \leftarrow t_{off}^{b}$, $t_{off}^{b} \leftarrow t_{off}^{a}$
12:         $t_{off}^{a} \leftarrow f_1(t_{off}^{s}, t_{off}^{e})$, $T_2^{\star} \leftarrow T_1^{\star}$, $T_1^{\star} \leftarrow PT(t_{off}^{a})$
13:     end if
14: end while
```

Algorithm 2 has the same input as Algorithm 1 except the accuracy coefficients ξ. The service bound and the upper bound of t_{off} are first derived (lines 1-2). Then the two endpoints of golden section selection are initialized as the lower and upper bounds of the feasible region of t_{off} (line 3). Lines 4-5 calculate the two tested points and their corresponding peak temperature with $PT(t_{off})$. Lines 6-14 purely do the golden section selection to discover the optimal t_{off} such that $PT(t_{off})$ reaches its minimum. Since golden section selection is a well known algorithm, the details are not addressed herein.

VI. Case Studies

In this section, we study the viability and efficiency of our algorithms and compare them with two approaches in [2], [11]. The simulations are implemented in Matlab (32 bit) using

TABLE I. Thermal and Hardware Model Parameters

G	C	$\varphi_i = \varphi_a$	θ_i	θ_a	T_{amb}	$t_{swon} = t_{swoff}$
0.3 $\frac{W}{K}$	0.03 $\frac{J}{K}$	0.1 $\frac{W}{K}$	-25 W	-11 W	300 K	0.1 ms

TABLE II. Event stream setting

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}
p (msec)	198	102	283	354	239	194	148	114	313	119
j (msec)	387	70	269	387	222	260	91	13	302	187
d (msec)	48	45	58	17	65	32	78	-	86	89
c (msec)	12	7	7	11	8	5	13	14	5	6

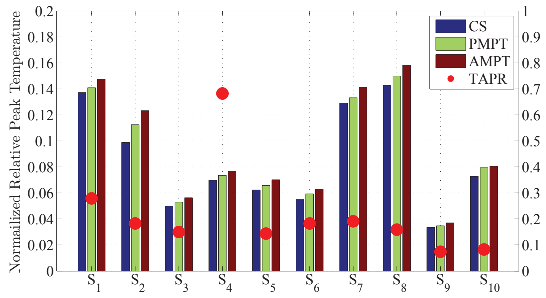

Fig. 6. Normalized Relative Peak Temperature produced by the tested approaches for single event stream scenarios with $\chi = 1$. The right Y axis indicates the NRPT of approach TAPR.

RTC-toolbox. All the results are obtained from a simulation platform with an Intel i7 4770 processor and 16 GB memory.

A. System Description

The thermal and power parameters are set as described in Tab. I [11], [19]. The task streams set studied in [18], [22] is used for our case studies and the parameters are summarized in Tab. II. Earliest deadline first (EDF) is adopted as the scheduling policy for multi-event scenarios. The (p, j, d, c) event model is adopted to specified an input stream S_i by its period p, jitter j, minimal inter-arrival distance d of the stream and the worst-case execution time c. Note that other common timing models of event streams can also be employed in our case studies with the concept of arrival curve. We choose the (p, j, d, c) model because it is a commonly used model and the arrival curve can be easily generated by an existing formula. The relative deadline D_i is defined as $D_i = \chi * p_i$ and varies according to the deadline factor χ.

The online approach cool shaper (CS) studied in [11], which relies on not only the upper arrival curve but also the actual arrivals of the coming events to dynamically shut down the processor, and the approach TAPR (thermal-aware periodic resources) studied in [2] are adopted for the comparison. The input event model used in TAPR is sporadic task (c, D, P), which is characterized by a worst-case execution time c, a (relative) deadline D and a minimum inter-arrival separation P. This model does not contain all the information of our (p, j, d, c) event model. Therefore, we revised P in a sporadic task as $\max[(p - j), d]$ to satisfy the worst-case deadline constraints. With these setups, Our algorithms are compared for both single and multi-event scenarios.

B. Simulation Results

First, we compare the minimal peak temperature derived by the four approaches. It is worth noting that the differ-

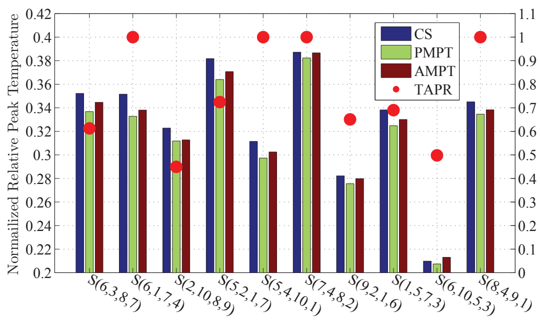

Fig. 7. Normalized Relative Peak Temperature produced by the tested approaches for ten sets of randomly selected four-events stream scenarios with $\chi = 1$ by applying EDF scheduling. The right Y axis indicates the NRPT of approach TAPR.

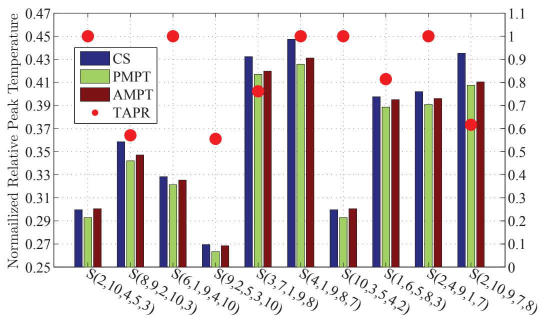

Fig. 8. Normalized Relative Peak Temperature produced by the tested approaches for ten sets of randomly selected five-events stream scenarios with $\chi = 1$ by applying EDF scheduling. The right Y axis indicates the NRPT of approach TAPR.

ences between the numerical values of those minimal peak temperature are hard to distinguish compared to their much larger absolute values. Thus the Normalized Relative Peak Temperature (*NRPT*), which is defined in the following, is employed as the index to evaluate the approaches:

$$NRPT_A = \frac{T^\star_A - T^\infty_s}{T^\infty_a - T^\infty_s} \tag{30}$$

where $NRPT_A$ and $T^\star_A$ is the Normalized Relative Peak Temperature and the minimal peak temperature produced by approach 'A', respectively. From its definition, a smaller *NRPT* indicates that the approach can better minimize the peak temperature.

Fig. 6 describes the *NRPT* for all the single event streams. Figures 7-8 reveal the results for four-events and five-events scenarios, respectively. Fig. 9 shows the derived minimal peak temperature w.r.t. different relative deadlines for the four approaches while taking all streams as input. Since the results of TAPR are much higher than those of the other three approaches, we display the results of TAPR with another Y axis in these four figures. Note that in multi-event scenarios, the arrival curves in CS must be approximated for EDF scheduling. Otherwise, the arrive curves will be too complicated and cause memory overflow for the JVM in Matlab [23].

From Figures 6-9, we state below observations. (1) In all these cases, approach PMPT generates better or no worse results than approach AMPT, this is expected because PMPT brutally searches all the possible solutions to get the precise t_{on} while AMPT relies on the approximate t_{on} to minimize the peak temperature. (2) For algorithms PMPT and AMPT, the minimized peak temperatures in four-events and five-event scenarios are much higher (*NRPT*s stay inside $[0.2, 0.45]$) compared to single event scenarios (*NRPT*s stay inside $[0.04, 0.16]$). This is caused by the fact that the processor has to handle more workload in multi-event scenarios and thence generates more heat. (3) As shown in Fig. 9, the peak temperature decreases as the relative deadline increases, since the processor can stay at sleep mode longer for each mode switch. The peak temperature however will not further decrease after certain threshold is reached. (4) The minimal peak temperature in CS is generally the lowest in single event stream scenarios as CS works online and can dynamically turn off the processor according to actual workload. It's worth noting that since the heat generated by online calculating

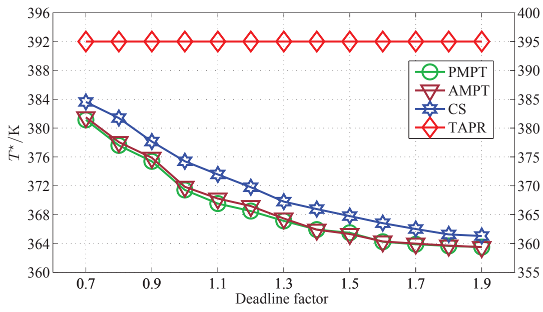

Fig. 9. Peak Temperature generated by the tested approaches w.r.t. different relative deadlines for ten-events stream scenario with EDF scheduling.

and monitoring of CS is not considered in our simulation, the peak temperature in CS will be higher when it comes to practical application. Moreover, CS approach also has to pay high penalty of the offline computation time while PMPT and AMPT approaches can achieve similar effect with much lower computation expense, which we will show later. In multi streams scenarios, however, CS yields higher peak temperature than PMPT and AMPT, which is resulted from the approximation of input arrival curves. We have made better approximations to improve the results but with trivial feedback. (5) By and large, the peak temperature derived by TAPR is the highest. The reason is the limitation of its event model where the non-determinism of pjd pattern cannot be properly modeled and the modified $P = \max[(p-j), d]$ overestimates the incoming workload. As shown in Fig. 6, there exists an extraordinary point, which is the *NRPT* of task S_4. The reason is that S_4 has the largest jitter j and the second smallest minimal inter-arrival distance d, which exacerbates the effect of the event model unsuitableness. Consequently, we can see that the peak temperature generated by TAPR reaches the upper bound in the multi-event cases as long as S_4 is involved in input streams.

We also report the timing overhead of deriving a PTM scheme. Since our PTM approaches are offline computed and need negligible runtime overhead, only the offline computing part of CS is taken into account. We adopt the computation time for finding the optimal W_{unit}, which is the critical parameter for CS, as the computational overhead of CS. Fig. 10 shows the computation expense of the four approaches for ten sets of randomly selected four-event streams and Fig. 11 demonstrates how the computation expense in ten-event stream scenario varies as the relative deadline factor changes. We make below observations: (1) The computation overhead of

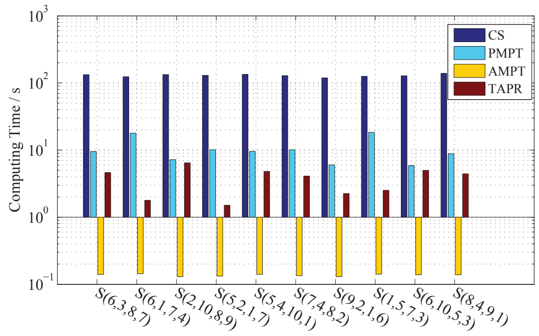

Fig. 10. Computing time of the tested approaches for randomly selected four-events stream scenarios with $\chi = 1$ by applying EDF scheduling.

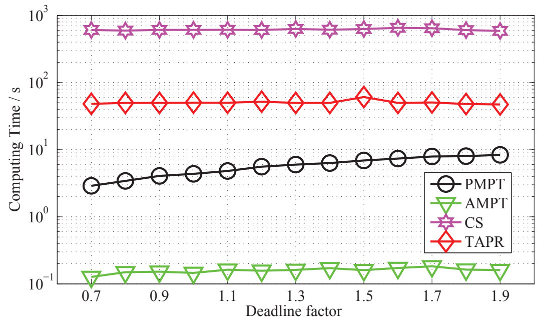

Fig. 11. Computing time of the tested approaches w.r.t. different relative deadlines for ten-events stream scenario with EDF scheduling.

cool shaper is the highest, which is about one up to four orders of magnitude larger than that of our PTM approaches. (2) In the second figure, the computation overhead of PMPT increases w.r.t. the relative deadline. The reason is that the number of the tested points of t_{off} and t_{on} increases as the relative deadline increases when ε and ξ are fixed. (3) The time consumptions of AMPT are always the lowest and stable, which are within half a second. (4) Compared to PMPT, the timing overhead of AMPT is about one or two orders of magnitude lower. In conclusion, our PTM algorithms are much faster in terms of computation overhead but generate peak temperatures close to or even better than the ones of CS online approach.

VII. Conclusion and Future Work

In this paper, we present new approaches to minimize the peak temperature for hard real-time systems in which the input event streams are characterized by arrival curves. With the worst case deadline constraints, we propose one approach that can provide precise solutions and one approach to yield approximated solutions with lower computation expense. To verify the effectiveness and efficiency, we present several implementations of our approaches with single event and multi-event streams. Experimental results show that our algorithms can derive periodic thermal management schemes with negligible runtime overhead while the peak temperature can be constrained to similar or even better level of online approach in the literature.

As topics of future work, the periodic thermal management can be further extended to hyper-period, which combines two or more patterns of PTM in one hyper-period. Moreover, we plan to implement our PTM in multi-core platforms and further demonstrate the benefits of using PTM approaches.

References

[1] C.R. Elks and et al. Reliability analysis of hard real-time systems in the presence of controller malfunctions. In *RAMS, 2000*, pages 58–64. IEEE.

[2] Masud Ahmed and et al. Minimizing peak temperature in embedded real-time systems via thermal-aware periodic resources. *Sustainable Computing: Informatics and Systems*, 1(3):226–240, 2011.

[3] David Brooks and et al. Dynamic thermal management for high-performance microprocessors. In *HPCA*, pages 171–182. IEEE, 2001.

[4] Sushu Zhang and et al. Approximation algorithm for the temperature-aware scheduling problem. In *ICCAD*, pages 281–288. IEEE, 2007.

[5] Thidapat Chantem and et al. Online work maximization under a peak temperature constraint. In *ISLPED*, pages 105–110. ACM, 2009.

[6] Shengquan Wang and et al. Reactive speed control in temperature-constrained real-time systems. *Real-Time Systems*, 39(1-3):73–95, 2008.

[7] Nikhil Bansal and et al. Speed scaling to manage energy and temperature. *JACM*, 54(1):3, 2007.

[8] Jian-Jia Chen and et al. Proactive speed scheduling for real-time tasks under thermal constraints. In *RTAS*, pages 141–150. IEEE, 2009.

[9] Pratyush Kumar and et al. Thermally optimal stop-go scheduling of task graphs with real-time constraints. In *ASP-DAC*, pages 123–128. IEEE Press, 2011.

[10] David Meisner and et al. Powernap: eliminating server idle power. *ACM SIGARCH Computer Architecture News*, 37(1):205–216, 2009.

[11] Pratyush Kumar and et al. Cool shapers: shaping real-time tasks for improved thermal guarantees. In *DAC*, pages 468–473. IEEE, 2011.

[12] Gang Chen and et al. Energy optimization for real-time multiprocessor system-on-chip with optimal dvfs and dpm combination. *ACM/TECS*, 13(3s):111, 2014.

[13] International technology roadmap for semiconductors. http://www.itrs.net/reports.html.

[14] Mark Benson. *The Art of Software Thermal Management for Embedded Systems*. Springer, 2014.

[15] L. Thiele and et al. Real-time Calculus for Scheduling Hard Real-time Systems. *ISCAS*, 4:101–104, 2000.

[16] Hettiarachchi and et al. A design and analysis framework for thermal-resilient hard real-time systems. *TECS*, 13(5s):146, 2014.

[17] Jean-Yves Le Boudec and et al. *Network calculus: a theory of deterministic queuing systems for the internet*. Springer, 2001.

[18] Kai Huang and et al. Periodic power management schemes for real-time event streams. In *CDC/CCC*, pages 6224–6231. IEEE, 2009.

[19] Devendra Rai and et al. Worst-case temperature analysis for real-time systems. In *DATE*, pages 1–6. IEEE, 2011.

[20] Kai Huang and et al. Applying real-time interface and calculus for dynamic power management in hard real-time systems. *Real-Time Systems*, 47(2):163–193, 2011.

[21] Gang Chen and et al. Energy optimization with worst-case deadline guarantee for pipelined multiprocessor systems. In *DATE*, pages 45–50. EDA Consortium, 2013.

[22] Ernesto Wandeler and et al. Optimal tdma time slot and cycle length allocation for hard real-time systems. In *ASP-DAC*. IEEE, 2006.

[23] Approximation of curves. http://www.mpa.ethz.ch/static/tutorial.html.

VIII. Appendix

We conducted various experiments to explore how the $T^{\star}$ in function 29 changes w.r.t the t_{off} in its feasible region. Since $\eta(t_{off} + t_{swon})$ is effected by the service bound, we first conducted a set of experiments to discover how different service bounds influence the function $PT(t_{off})$. In these experiments, thermal coefficients m_a and m_s are assumed to be the same because the difference between them is not the factor we mainly concerned. We conducted another set of experiments to reveal how this difference affects the results. The time overheads of switching the system on and off are assumed to be same in all experiments.

TABLE III. THE RANGE OF INPUT EVENT STREAMS AND SYSTEM MODEL COEFFICIENTS IN THE FIRST SET OF EXPERIMENTS

	p (msec)	j/p	d/p	c/p	χ	$t_{swon} = t_{swoff}$(msec)	$m_a = m_s$
from	20	0.1	0.1	0.05	1	0.1	1
to	1000	1.3	0.1	0.7	2	4.1	20
step	20	0.3	-	0.05	0.2	0.5	1

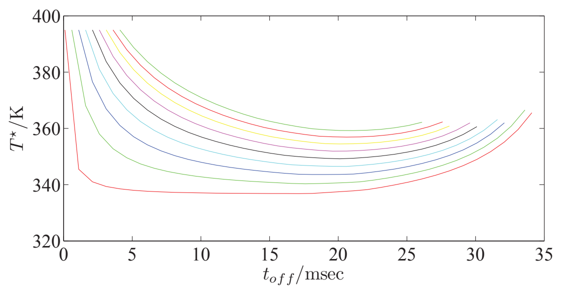

Fig. 12. Peak temperature of the processor for different transition overheads with event stream period $p = 20ms, c = 3ms, j = 2ms, \chi = 2$ and $g_a = g_s = 6.77$.

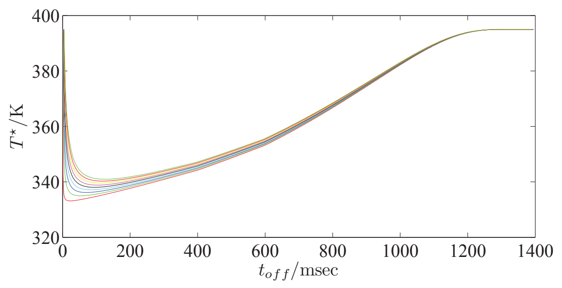

Fig. 13. Peak temperature of the processor for different transition overheads with event stream period $p = 1000ms, c = 100ms, j = 100ms, \chi = 1.5$ and $g_a = g_s = 6.77$.

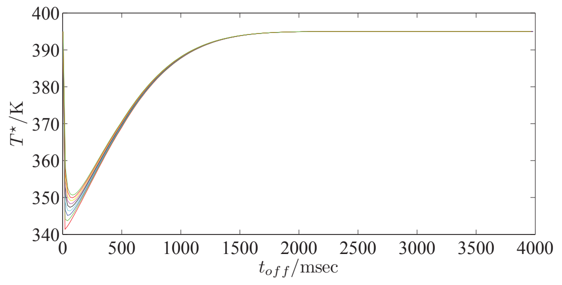

Fig. 14. Peak temperature of the processor for different transition overheads with event stream period $p = 5000ms, c = 1000ms, j = 500ms, \chi = 1$ and $g_a = g_s = 6.77$.

For the first set of experiments, the system model and input streams parameters are depicted in Tab. III. Based on the setup, 3780000 scenarios have been tested in our experiments. In every scenario, the function $PT(t_{off})$ is sampled with a step $\varepsilon = 1ms$ in the feasible region of t_{off}. Then the sampled points are used to determine if the function $PT(t_{off})$ in current scenario is unimodal. The results demonstrate that $PT(t_{off})$ only has one minimum in the feasible region of t_{off}. Due to space limit, we only offer part of the results in the following figures. In the first three figures, there are nine curves in every figure. Each of them represents the peak temperature varies as t_{off} changes with an individual mode-switching overhead in $\{0.1, 0.6, 1.1, \cdots, 3.6, , 4.1\}$(msec).

Based on Tab. IV, we set up the second set of experiments to discover if $PT(t_{off})$ is still unimodal when m_s doesn't

TABLE IV. THE RANGE OF INPUT EVENT STREAMS AND SYSTEM MODEL COEFFICIENTS IN THE SECOND SET OF EXPERIMENTS

	p (msec)	d/p	c/p	χ	$t_{swon} = t_{swoff}$	m_a	m_s/m_a
from	20	0.1	0.05	1	0.1	1	0.5
to	1000	0.1	0.7	2	4.1	20	1.5
step	20	-	0.05	0.2	0.5	1	0.1

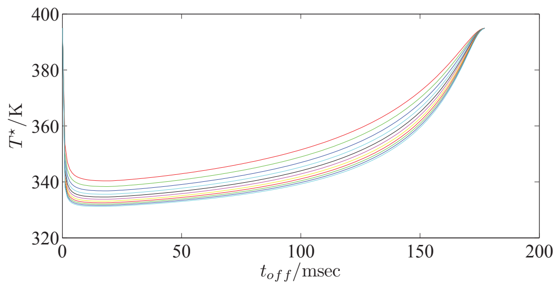

Fig. 15. Peak temperature of the processor for different thermal coefficients with event stream period $p = 200ms, c = 20ms, \chi = 1, t_{swon} = t_{swoff} = 0.6$ and $m_a = 5$.

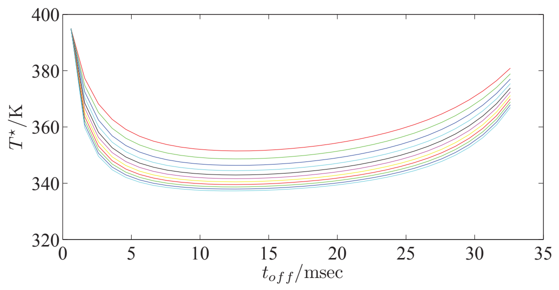

Fig. 16. Peak temperature of the processor for different thermal coefficients with event stream period $p = 40ms, c = 4ms, \chi = 1, t_{swon} = t_{swoff} = 0.6$ and $m_a = 8$.

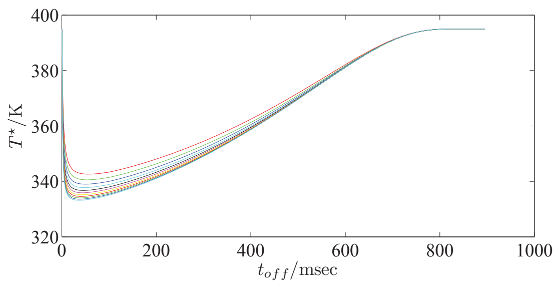

Fig. 17. Peak temperature of the processor for different thermal coefficients with event stream period $p = 100ms, c = 100ms, \chi = 1, t_{swon} = t_{swoff} = 0.6$ and $m_a = 8$.

always equal m_a. The jitter factor j/p is fixed as 0.1 in the experiments. Part of the results are presented in the following three figures. In every figure, there are eleven curves and every curve demonstrates the relationship between peak temperature and t_{off} for an particular m_s/m_a ranging from 0.5 to 1.5 with a step of 0.1 Each of them represents the peak temperature varies as t_{off} changes with an individual mode-switching overhead, ranging from 0.1 to 4.1 with a step of 0.1.

As shown in the figures, the peak temperature first decreases and then increases as t_{off} grows, which is an unimodal function based on the definition. The remaining results of our experiments have the same conclusion. Therefore, we conjecture that $PT(t_{off})$ is an unimodal function.